A processor for testing mixed-signal cores in System-on-Chip

Francisco Duarte, J. Machado da Silva, José C. Alves, G. A. Pinho, José S. Matos
Faculdade de Engenharia da Universidade do Porto, INESC Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
{fduarte, jms, jca, gabriel, jsm}@fe.up.pt

Abstract

This paper describes the design of a processor specific for testing cores embedded in system-on-chip. This processor, which can be implemented within a system's reconfigurable area, shall be responsible for scheduling and control test operations and perform preliminary data processing, as well as to provide the interface with an external tester. Building these test operations on-chip allows for simplifying external tester interface and to reduce testing time. The testing procedure and the infrastructure required to test an A/D converter is described as an example.

1. Introduction

The progress attained in successive generations of system-on-chip (SoC), has created a new range of innovative and affordable consumer products. Blocks such as digital and analogue I/O interfaces, complex communication sub-systems (including optical and radio-frequency circuits), power management, and multiple processors with the respective software, can now being integrated onto single silicon dies.

Conventional test approaches, fully relying on external automatic test equipment (ATE), are unable to cope with the test requirements of tens or even hundreds of such cores deeply embedded in complex systems. This is particularly true for analogue and mixed-signal (AMS) SoCs found in the markets of wireless and wired communications, along with consumer electronics products.

To overcome this drawback specific built-in self test (BIST) schemes may be used in order to simplify the interface with the external tester, performing parallel tests of different cores, and generating specific test stimuli through on-chip processing. Such approaches shall always be developed having in mind the objective of reducing test time and cost.

1.1. Test processors

The use of processors to perform different test operations has already been proposed. These include self-testing [3], memory tests [10], and the entire test of a SoC [6]. Both hardware and software specific facilities can be provided in these processors, such as boundary-scan controllers [9], Linear Feedback Shift (LFSR) and Multiple Input Shift (MISR) registers, and programs for local test vector compression and decompression [2]. In [1] an embedded AMS test controller is proposed which makes use of the IEEE 1149.4 standard [8] and utilizes the embedded memory to support test operations.

The solution proposed here relies on reusing the logic reconfigurable block (such as a field-programmable gate array – FPGA) existing in a SoC to perform some other mission function, to implement an application specific instruction set processor (ASIP) to control and schedule on-chip test operations. This processor can be adapted to the specific test needs of each block under test, in order to fully exploit the reconfigurable resources available within the system and minimize the test time. For example, different test mechanisms are required for an A/D converter, a RAM block, or a bank of digital filters, and the requirements of each test depend whether the test is intended for production or field maintenance. Furthermore, depending on the architecture of the SoC under test, the test processor may also include support to manage the SoC test infrastructure and configuring the routing of normal mission and test signals. Being an instruction-set processor implemented on a reconfigurable platform, a very flexible architecture can be obtained that is enhanceable with dedicated instructions tailored to meet application specific test needs.

The rest of the paper describes, in section 2, the infrastructure and the test processor main functionalities being proposed. Section 3 describes the processor's architecture and its instruction set. An application example for the specific case of an ADC core is presented section 5. Section 6 highlights the main conclusions.

2. The test processor and test infrastructure

The architecture of the test infrastructure and test processor to implement and manage analogue and mixed-signal test operations is presented in this section. The test processor is built as a RISC style load-store processor core with a customizable instruction set, surrounded by modules that handle test-specific tasks supported by custom instructions.

2.1. Test Infrastructure Control

The type of interface implemented between the tester and the core under test (CUT) depends on the test resources included in the circuit. Increasing the amount of in-circuit test resources allows increasing controllability and observability. On the other hand, one has to distinguish between mission and test signals. Generally, test signals include dynamic and static signals. The former ones are active or change often during testing, while the seconds are typically those which are only used to configure test modes or do not change often during test [11]. While these signals can be propagated and configured using serial test infrastructures, the dynamic ones need parallel routing.

A standard boundary-scan IEEE 1149.1 serial chain, or a generic scan chain, can be used to propagate static signals and to switch between mission and test functional modes. An infrastructure controller resident in the processor provides this facility. Furthermore, reconfigurable logic allows also to implement IEEE 1149.1 boundary-scan (or other type) scan cells to route signals from cores which are not provided with test wrappers. All these registers shall be controlled by the test processor.

As an example, figure 1 illustrates the case of the CUT being an A/D converter and the test processor placed in the FPGA. This structure represents the generic case of the CUT being a peripheral within the system. The ADC wrapper includes the cells required to switch between mission and test signals (either dynamic or static). In general, mission signals are connected to other blocks in the system, and test signals are connected to the FPGA. When necessary, additional Test Control Mode (TCM) and Test Multiplexer (TMX) registers provide accessibility for additional test signals, coming from external or internal resources. This can be the case of the Test Bus Interface Circuit (TBIC) of the IEEE 1149.4 analogue test bus [8], which could be used to drive analogue test stimuli coming from the external ATE. Analogue test cells placed in the core's wrapper allow for, besides switching between mission and test signals, a single internal analogue test bus to drive stimuli to different cores.

Dynamic input signals require that two input cells be present in the wrapper to switch between system mission signals and the test ones generated by the test processor.

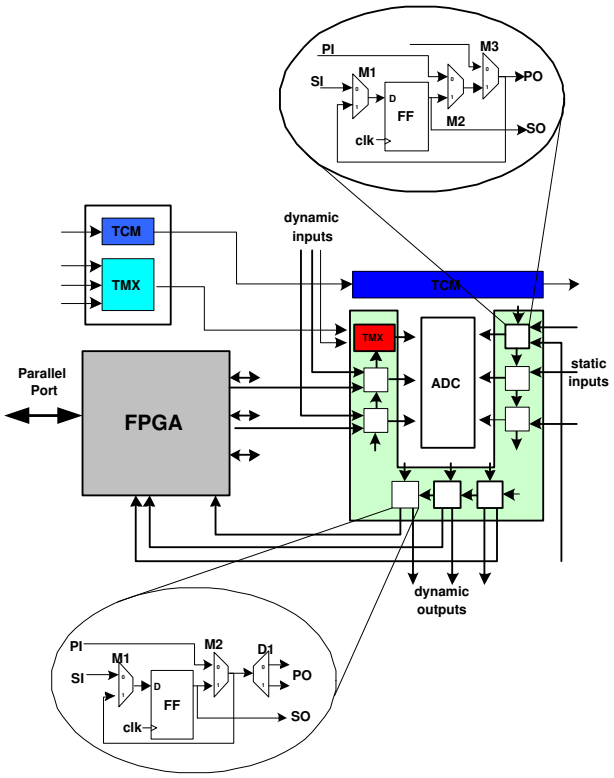

Figure 1. Example of interface between the CUT and the test processor.

Similarly, the CUT dynamic outputs require cells capable of directing these signals to mission lines or to the test ones that connect directly to the test processor. These cells are similar to the ones presented in [7] but include both serial and parallel test inputs and outputs. The FPGA offers thus also the possibility of implementing the digital part of test cells, in a more flexible way as it would be easier to design these cells following the requirements of each specific case.

Another aspect concerns time control and synchronization of analogue and digital events. In analogue test it is often required that sampling (for stimulus generation and response capture) be performed at a constant rate. In general, the analogue frequencies are smaller than the digital clock frequencies used in the test processor, however, it is critical that during stimuli generation and response capture, the processor does not introduce fluctuation in sampling frequencies. This aspect will be developed further later in the paper.

2.2. Test stimuli generation and response capture

The variety of test operations to be performed in an integrated system (interconnects, modules integrity, overall functional performance) requires different test stimuli, observation, time control and response evaluation schemes. Concerning stimuli generation, besides the possibility of choosing among different test stimuli generators, such as Pulse Width Modulator (PWM), LFSR, Direct Digital Synthesizer (DDS), it is also important to assure a high degree of inter-operability between the stimulus generator and the core processor to allow a flexible stimulus waveform amplitude and frequency definition. On the other hand, it is fundamental that the stimulus generation, after being programmed, be functionally autonomous to avoid requiring a permanent interaction with the core processor, freeing it for other operations, for example capture and pre-processing of the response. A digital sigma-delta modulator implemented in the FPGA allows to generate analogue signals requiring a simple RC network to filter the 1 bit modulated FPGA output signal.

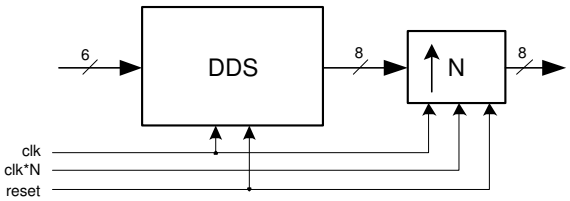

Figure 2. Stimuli generation with DDS and interpolation.

Figure 2 illustrates the block diagram of a stimulus generator comprising a DDS and an interpolator. The combination of these two blocks allows to find a good compromise between the area occupied in the FPGA, and the generated stimulus resolution. Using a 8-bit DDS, and a 2 times interpolation (N=2), allows to obtain an equivalent higher resolution, with a shorter look-up table (LUT). The interpolator assigns values between consecutive LUT words, and thus increases the sampling rate of the D/A process, leading noise (image frequencies) in the output spectrum to be shifted to higher frequencies. For a certain output filter bandwidth, this has the effect of pushing noise out of band, thus reducing the in-band noise and increasing the equivalent resolution.

Figure 3 shows a detailed diagram of the DDS. In this case, to generate a sinusoidal stimulus, only 90º of the waveform are stored in the LUT, being the blocks “1’s compl”, “-1” (2’s comp), and the output multiplexer used to generate the full cycle after this sub-set of the waveform. The stimulus frequency is defined by the ratio between “clkDDS” and “enDDS” clock signals.

Testing time is a critical issue in production test. Although not expected to be a frequent case, aborting a test operation when catastrophic faults are detected avoids unnecessary further test operations. For this purpose, capture instructions include the possibility to generate an interrupt in case the captured sample presents a definitely erroneous value. For each captured sample, if the variation from the previous one is higher than a specified expected value, it is assumed that the CUT’s response presents a totally non-admissible behaviour.

Power consumption in testing can be problematic in SoCs, as usually the system’s power management is designed considering only the normal operation mode, and also because different test operations may be run in parallel to reduce test time. To reduce power consumption the processor’s clock frequency can be adjusted to the specific requirements of each test stage. For example, during stimuli generation and response capture the processor may run with the lowest clock frequency that can guarantee the required sampling frequency, but to pre-process the acquired response an higher speed would be convenient to reduce processing time.

2.3. Preliminary processing operations

Performing preliminary test data processing on-chip, allows reducing the volume of data to be transferred to the external tester, as well as the post-processing needed to obtain the final test results. This contributes also to reduce testing time, both on transferring and post-processing time. Depending on the testing methods being used, different processing operations may be required. Besides those which can be performed with the ALU included in the test processor, more specific operations can be carried-out using additional test logic or within auxiliary blocks that can be added, exploring the execution of parallel operations.

One of the possible processing operations to be implemented is the cross-correlation of the captured response with in-phase and quadrature forms of the test stimulus, in order to calculate gain, phase, and harmonic distortion. The block ”+PI/2” shown in figure 3 is used to generate the quadrature signal. Re-generating the test stimulus in processing operations avoids the necessity for storing it when it is applied to the CUT.

3. Processor’s architecture

To make use of an FPGA area inside a SoC for testing one part of the SoC, it is necessary to design a custom digital circuit to execute the appropriate tasks in the correct se-

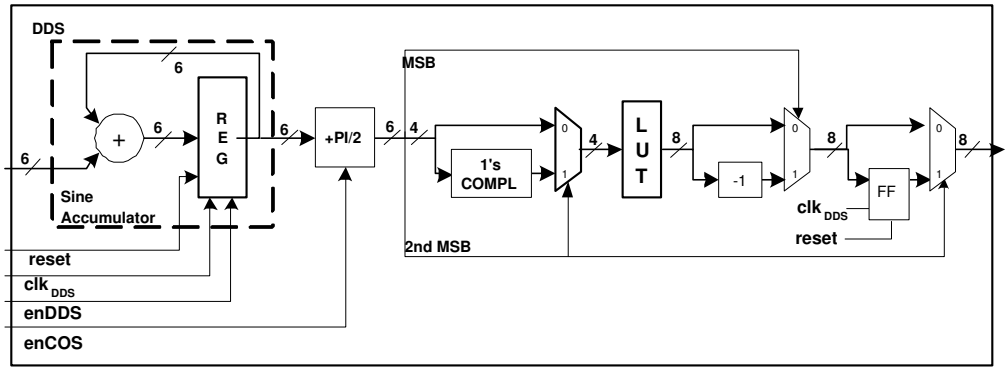

Figure 3. DDS block diagram.

quence. This involves four main operations: controlling the test infrastructure to gain access to the part to be tested, generate a set of stimuli and apply them to the part under test, capture the output values and, finally, process the captured results.

All these operations can be efficiently undertaken by custom designed digital circuits, relying on traditional digital design techniques. However, using such development methodology to design the different circuits required for the various cores to be tested and test stages is time consuming, error prone, and inflexible.

The approach proposed here resorts on obtaining Application Specific Instruction-Set Processors (ASIP), whose set of conventional and dedicated instructions are automatically derived from a software specification of the test operation to be implemented. Figure 4 shows a block diagram of the test processor. It includes a programmable processor core surrounded with several test specific functional blocks, which may be included or not, depending on the type of test operations to be performed.

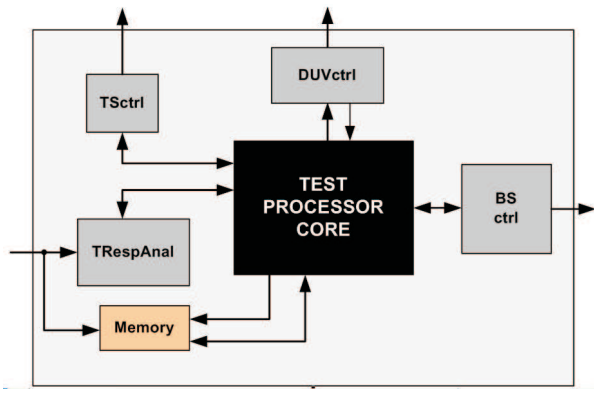

Figure 4. Test processor architecture.

The base processor core supports a complete set of conventional instructions, plus all the specific instructions that control the test-specific functional blocks placed around the core. In this case we have:

- **BSctrl** is a boundary-scan controller implementing the IEEE 1149.1/.4 standard protocol, meant to control the 1149.1 test infrastructure;
- **TSctrl** is a block dedicated to stimulus generation
- **DUVctrl** is a module responsible for controlling the operation of the CUT;
- **TrespAnal** is dedicated to evaluate the validity of the samples being captured;

The actual configuration of the test processor is determined by the instructions the designer uses in the test program, and includes only the resources required for that task. For example, if the processor's registers and the ALU operations are not referred in the program to be executed, these elements do not need to be included in the processor's datapath and control path; in the other hand, if a sine waveform generator is needed to stimulate an analogue block, a special instruction must be used that will attach to the processor core a block that support this function. Section 4 presents the procedure to create a configuration of the test processor from the textual specification of a test program.

Although this strategy creates programmable processors that do not exhibit a high degree of flexibility because they only include the resources to meet the needs of one particular program, it is an efficient way to create dedicated controllers area-optimized for each particular test task. Besides, any test programs that use the same set of instructions can still be implemented without requiring the re-synthesis of a different processor.

3.1. The instruction set

Besides conventional load/store, arithmetic/logic operations and loop control, the instruction set comprises application specific instructions supported by peripheral blocks, to handle the IEEE 1149.1/4 port that controls the CUT wrapper, the test stimuli generator, and to handle the CUT specific control signals:

- **SRTEST** Starts the test procedure by applying appropriate control signals to the device under test;
- **NSHF TDIdata** Load the 1149.4 infrastructure registers with a specific bit stream;
- **STATE TMSdata** Changes the TAP controller state machine to the specified state, applying the appropriate number of TCK ticks and TMS values;
- **STIMULUS** Applies a digital stimuli vector previously stored in a specific memory segment to the **TSctrl** stimuli generator block;
- **EOTEST** Signalizes the end of test and resets internal memory pointers to prepare for another test;

The instructions SRTEST and EOTEST are meant to be used to support particular CUT functions under the control of the DUVctrl module. The generation of test stimuli is performed with the STIMULUS instruction, which acts over the TSctrl block. Once that instruction is executed, the generation of the stimulus signal is functionally independent from the core processor.

This set of instructions was designed in order to enable the parallel execution among them. For example, while the processor waits a new test response sample, it can evaluate the validity of the previous sample captured. This parallelism can also be exploited when the processor waits for an external synchronization, to perform some specific operation related with the CUT.

4. Test processor configuration

To customize the test processor for a specific test program, an application was developed that configures the various parts of the processor in order to include only the resources necessary to run that test program. The output of this process is the complete RTL model of the test processor, organized as a set of synthezisable HDL modules, that can enter the specific FPGA synthesis and implementation tools (figure 5).

The starting point for this customization is the source code of a test program, which may refer any instruction available in the full instruction set. The optimization procedure reads the source code and extracts a list of the instructions used, as well as the registers and ALU operations referenced. For each instruction, its description is extracted from a database that defines the sequence of states

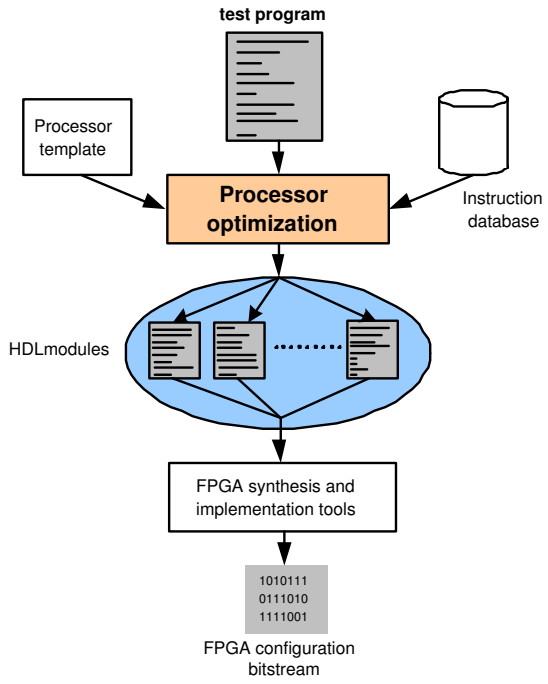

Figure 5. Design flow to configure and implement a custom test processor.

for the control path and additional blocks that may have to be included in the datapath to support the execution of that instruction. The set of registers referenced in the program configures the processor's register file, and the group of arithmetic and logic operations used in the code select the functional blocks to be included within the ALU. Finally, the finite state machine that implements the control path is adjusted to minimize the number of states and a predefined state encoding is assigned to the final HDL model.

Although some characteristics of the processor are fixed (for example the width of internal data buses and the encoding of instructions), this configuration procedure leads to significant reduction to the occupation of the FPGA, when compared with the complete version of the processor. If the target platform is a FPGA area with a large amount of configurable resources that can support the whole processor, it is not worth the effort to reduce the area of the test processor. In this case, a test procedure composed of various sub tasks (e.g. configure devices, apply stimuli, analyze results) may be implemented by only one instance of the test processor. In the other hand, if the reconfigurable area is small and can only implement subsets of the processor, a test procedure may be split into different stages, each one accomplished by a different version of the test processor, reconfiguring the FPGA area as many times as necessary. In this sit-

Processor model	4-input LUTs	Flip-flops
Full processor	1575	749
Cross-correlation	821	540
Dithered step wave	418	257

Table 1. FPGA occupation for different test processor configurations.

uation, the test designer will only write the test programs for each stage, and a specific processor will be created to run each one of those programs.

Table 1 shows the FPGA occupation for three different configurations of the test processor. These implementations where done with Synopsys FPGA Express for a XC4013 FPGA. First row refers to the full version that supports the whole instruction set, peripheral blocks and ALU operations. The second configuration was obtained for the specific case described in next section. The last version corresponds to a different specific architecture that applies as stimuli a dithered step wave, captures a set of samples and computes the average of responses in each step [5]. As one can see, there is a significant variation of the FPGA occupation when the processor is configured to include only the resources required for the specified operations. If an FPGA block is reconfigured twice to implement the two test procedures referred in table 1 it will require only 52% of the logic resources that would be necessary for the complete processor.

5. Application example

The principles and design philosophy presented before has been developed and used to develop a test strategy for an electronic energy meter system. Besides the central processing unit this system comprises in the signal acquisition front-end an FPGA dedicated to control and capture the digitised signals from the A/D converter. The system specifications require that the system performs a monthly test to check accuracy of captured data. Figure 6 shows the diagram of the prototype built to evaluate the test processor and test strategy of the data acquisition front-end. In this case a 1149.4 test bus is used to drive analogue signals from the stimulus generator output to the ADC input, as well as to the input impedance adapter (not shown in the picture) placed before the ADC. The analogue test cells are implemented with a commercial IC that implements the 1149.4 infrastructure. (The use of the this standard infrastructure is not mandatory and a simpler bus could be used, however it was decided to implement it here as a demonstration prototype.)

The test to be carried-out consists on applying a sine wave stimulus, and performing the cross-correlation of the captured response with in-phase and quadrature versions of the stimulus. Gain, phase, and harmonic distortion can be obtained after these two correlations [4]. The following script shows the main processor's program to perform the test operation.

```
NHARMONICS .EQU 5 ;no. of harmonics to determine
 SETDDS        ;DDS as stimuli generator
               ;sinusoidal signal
               ; Set test infrastructure
 STATE 11,447
 NSHF 39,1048575
 STATE 5,7
 NSHF 48,4194561
 STATE 2,3
 MOVC CNT2,NHARMONICS
               ;start test execution
 STIMULUS
 TEST_CS5330A  ;schedules DUVctrl CUT block
L: XCORR  RAD  ;performs cross-correlation
 XCORR LAD     ;performs cross-correlation
 STIMULUS
 DJNZ CNT2,L   ;go to next harmonic
 STATE  6,3F   ;go to Test-Logic-Reset state
 EOTEST
 HLT
.END
```

The first instruction specifies the DDS block for stimuli generation. The next 5 instructions interact with the "BSctrl" module to control the test infrastructure. The configuration of the infrastructure test mode is done by selecting the appropriate number of test clock (TCK) ticks and the test mode select (TMS) value to apply, and the bitstream to program the test infrastructure registers. These instructions perform the appropriate sequence to up-load the infrastructures registers with the required content [8].

After the test infrastructure has been configured the number of harmonics to be calculated is specified. The STIMULUS instruction controls the "TSctrl" module to generate the test stimulus previously specified. This stimulus is generated with the DDS block described before followed by a $\Sigma\Delta$ modulator and low-pass filter which provide the digital to analogue conversion. With the stimulus signal being applied to the CUT, the TEST-CS5330A instruction starts the ADC clock and the time delay required for the ADC initialization. This instruction is similar to SRTEST but specific for the ADC under test. The end of this initialization is signaled by the "DUVctrl" block. This time is also used to let the CUT transient response to settle-down. Data capture is also performed within the TEST-CS5330A instruction. The number of samples to be captured has been previously stored in the "DUVctrl" module.

The XCORR instruction activates the cross-correlator auxiliary block to perform this operation (the two XCORR instructions refer to right and left ADC channels). The ALU can also be involved in this operation under the control of the cross-correlation block. Upon the test completion the TAP state machine is brought to the reset state (STATE in-

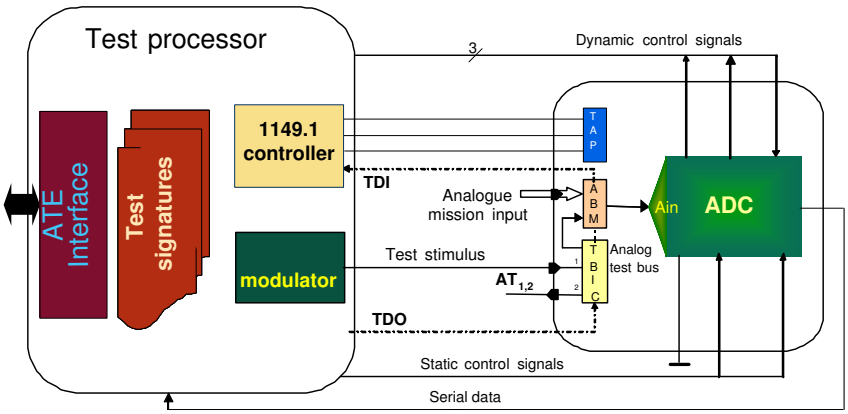

Figure 6. ADC test example.

struction) and the HLT instruction is executed to signal that the operation is concluded.

Figure 7 shows the stimulus spectrum at the ADC input. It is a 50 Hz sinusoidal wave with 4 Vpp amplitude. It can be seen that some harmonic content is present at low frequencies. Although their relative amplitudes are small, these can affect test results if very low total harmonic distortions are being measured. It was found that their presence is due to fluctuations in the amplitude of $\Sigma\Delta$ bit stream at the output of the FPGA. Table 2 shows average values of the harmonics obtained after performing the test operation. These values reflect both the distortion introduced by the data acquisition front-end, as well as the residual one from the stimulus. This might lead to the necessity of inserting a buffer to adapt the FPGA output to the load being driven.

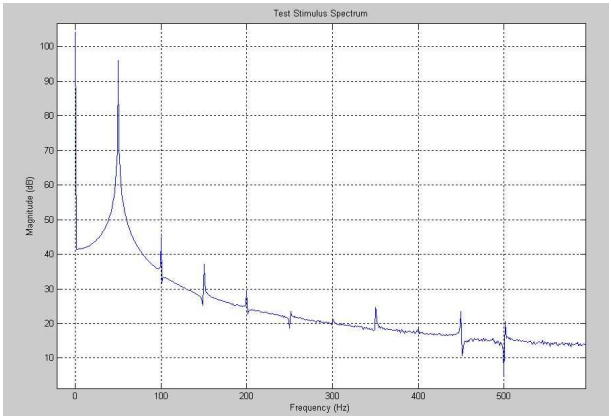

Figure 7. Stimulus spectrum.

H_n/H_1	amplitude (dB)
h2	-52
h3	-55
h4	-60
h5	-70

Table 2. Amplitude of harmonics obtained in the test operation.

6. Conclusions

A processor specific for testing embedded cores in a SoC is described in this paper. It provides on-chip stimuli generation, test control and scheduling, the control of a compliant IEEE 1149.1/4 test infrastructure, and data processing. These facilities allow reducing the number of signals to be sourced by the external tester, the amount of data to be transported between the tester and the CUT, as well as test time by performing on-chip pre-processing operations. The solution proposed here relies on reusing an FPGA-like block that may exist within the SoC to implement an application specific instruction set processor, that can be scaled according the test needs and the space available for implementation.

Having a programmable processor dedicated to this task and implemented in a digital reconfigurable FPGA-like block has several advantages, when compared to test-specific controllers with fixed functionality. First, there is no significant silicon area overhead, because the reconfigurable platform where the test processor is implemented already exists in the SoC. Second, depending on the size of the FPGA block available in a particular system, an adequate version of a test processor may be chosen, in order to offer a convenient trade-off between processing power and FPGA occupation.

Acknowledgment

The work presented herein has been partly supported by the Portuguese government - Agência de Inovação under the framework of projects ASSOCIATE (A503 - MEDEA+) and NanoTEST (2A702 - MEDEA+, phase 2).

References

[1] M. AbdEl-Halim. An analogue mixed-signal test controller. In *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, 2002.

[2] J. Abhijit and N. A. Touba. Deterministic test vector compression/decompression fos systems-on-a-chip using an embedded processor. *Journal of Electronic Testing - Theory and Applications, Special Issue on SOC (System-on-a-chip) Testing for Plug and Play Test Automation*, 18(4/5):503–514, August/October 2002.

[3] F. Corno, M. S. Reorda, M. Squillero, and M. Violante. On the test of microprocessor ip cores. In *Proceedings of the IEEE Design Automation and Test in Europe*, pages 209–213, March 2001.

[4] J. M. da Silva, J. S. Duarte, and J. S. Matos. Functional in-circuit characterisation of $\Sigma\Delta$ modulators. *Measurement, Journal of the International Measurement Confederation IMEKO, Elsevier, Special Issue on ADC Modelling and Testing*, 32(4):257–264, November 2002.

[5] F. X. Duarte, J. M. da Silva, J. C. Alves, and J. S. Matos. An infrastructure and application specific processor for testing analogue and mixed-signal socs. In *XIX Conference on Design of Circuits and Integrated Systems*, November 2004.

[6] P. Galke, C. and H. T. M., Vierhaus. A test processor concept for systems-on-a-chip. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, September 2002.

[7] IEEE P1500 Working Group. IEEE p1500 standard for embedded core test (sect). http://grouper.ieee.org/groups/1500, 2003.

[8] Mixed-Signal Working Group. *IEEE 1149.4 - Standard for a mixed-signal test bus*. Test Technology Technical Committee of the IEEE Computer Society, New York, NY, 1999.

[9] J. S. Matos, J. M. Ferreira, and F. S. Pinto. A boundary scan test controller for hierarchical bist. In *Proceedings of the IEEE International Test Conference*, October 1992.

[10] R. Rajsuman. Testing a system-on-a-chip with embedded microprocessor. In *Proceedings of the International Test Conference*, pages 499–508, October 1999.

[11] G. Seuren and C. Feige. Extending core test methodology to the analogue/mixed-signal domain. In *Proceedings of the IEEE European Test Workshop*, May 2001.